Software Engineering Fundamentals

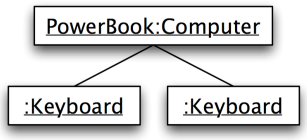
- Lecture 3 : Class, Object diagrams

Modelling structure

- The problem domain is the topics/concepts that the client deals with.
- We can contrast it with the solution domain - the topics/ concepts that we deal with to provide an implementation for the client.
- The domain model is the structural model of the domain - in UML it consists of a class diagram.
- A class diagram consists of a set of classes and the relationships between those classes.

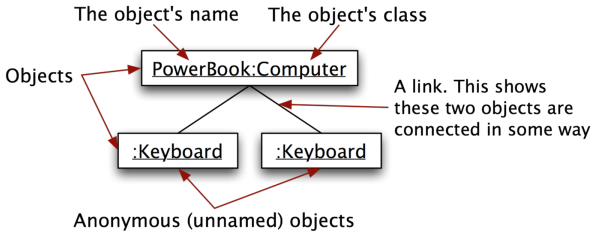
Object diagrams

- An object diagram shows the associations between real or possible objects that we are modelling.



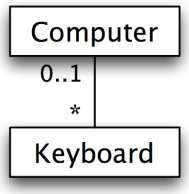
Object diagrams

- An object diagram consists mainly of objects and links.
- Objects are instances (occurrences) of a specific class or grouping.
- Links are examples of relationships between objects.



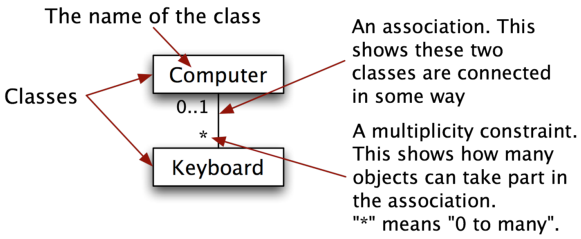
Class diagrams

- A class diagram is an abstraction of all the possible object diagrams that we are interested in modelling.



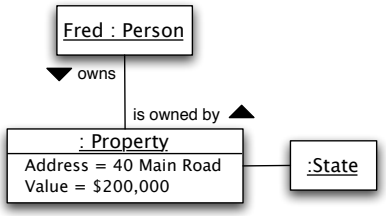
Class diagrams

- An class diagram consists mainly of classes and associations.
- Classes are generalisations of the objects being modelled.
- Associations are a generalisation of a link.



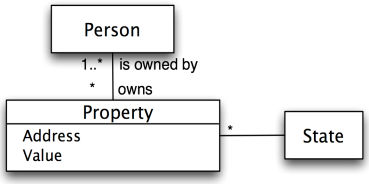
More on object diagrams

- This object diagram shows that Fred owns a property.
- If Fred bought another property we would have to modify this diagram.
- Under what circumstances would we have to change the link between the Property object and the State object?
- Here we have named the roles that each object plays in the link between Person and Property.



More on class diagrams

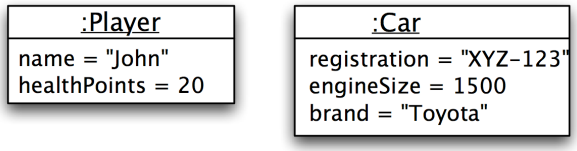
- The class diagram for the previous example is



- The multiplicity constraints state that...
  - A property must be owned by one or more people (1..*)
  - A person can own 0 or more properties (*)
  - Every property must be in a State
  - A state can must have one or more properties (1..*)

Object diagrams

- Objects can show attribute values
- This is useful when you want to ensure the modeling covers the needed features.



Attribute types

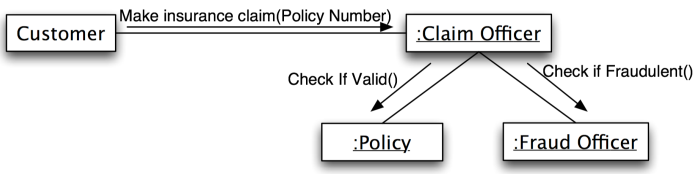
- Common attribute types are
  - Numbers
    - Integers - Natural, Positive, Real
  - Boolean - True False
  - Text
  - Enumerated types (e.g. {north, east, south, west})
  - Character (Y, N) - less commonly used
  - Date
- What attribute types would you use for
  - Age
  - Sex
  - Phone number
  - Land area
  - Global coordinates

How are classes/objects identified?

- Candidate objects/classes are identified by inspecting the use case diagrams (and iteratively via use case text format)
- Consider the use case text presented previously...

Object diagrams

- An object diagram can detail an existing business's work processes and relationships.
- This will be explored in later lectures



Class diagrams

- Class Diagrams model the static design view of the system.
  - Although the associations between objects can be dynamic, the types of associations are constrained by the class diagram.
- Classes contain…
  - classes, class names, class attributes
  - operations/methods
  - notes and constraints
  - many other features.
- Classes are connected to each other by relationships.
  - Relationships can be Associations or Generalisations.
- Class diagrams can also contain
  - notes and constraints.

Class diagrams - Multiplicity

- When you examine an object diagram (or the underlying problem you are modeling) you can determine how many objects are involved in each type of associations.
- We include this information which is known as Cardinality or Multiplicity on a class diagram.
- These document the constraints on the associations that we expect a program to enforce.
- Cardinality is:
  - * - 0 .. infinity
  - 2..4 - a restricted range
  - 1, 3..4, 6..* - comma separated lists
  - <empty> - implies 1, i.e. a mandatory connection

Class diagrams - Multiplicity

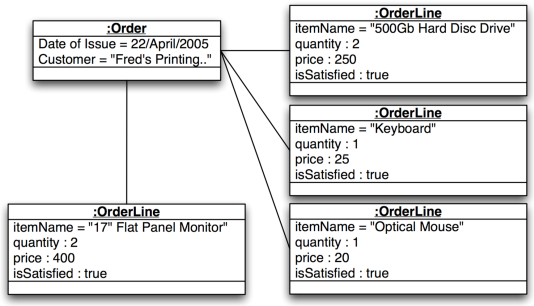
- Consider the following order form that may be collected during requirements.
- It shows an order system for a business/customers
  - each order consists of several lines e.g.

ABC Computer Supplies
Order Form
Date of issue 22nd April 2005
Ordered by : Fred's Printing Service

Item	Quantity	Unit Price
500Gb Hard Disc Drive	2	$250
Keyboard	1	$25
Optical mouse	1	$20
17" Flat Panel Monitor	2	$400

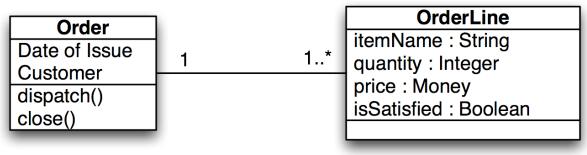
Class diagrams - Multiplicity

- To identify the multiplicity, you can examine an object diagram.



Class diagrams - Multiplicity

- The multiplicity constraints state that…
  - every OrderLine object must be associated with an Order
  - every Order must be associated with at least one, but potentially infinite OrderLines.



Draw class diagrams for

- A house has multiple rooms
- A person can own properties on their own, and with other people.
- Companies can own properties

How do we check if they are correct?

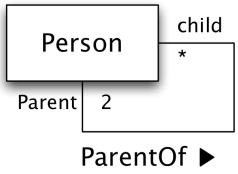
- You must always test your class diagrams.
  - Identify scenarios that you know are valid
  - Check that the model supports them

Class diagrams - Self association

- Self associations is when a class has an association with itself
  - Associations with other objects of the same class
    - A Parent/Child self association on class Person
  - Associations between one object and itself.
    - A Doctor/Patient association, where a doctor treats herself.

Class diagrams - Self association

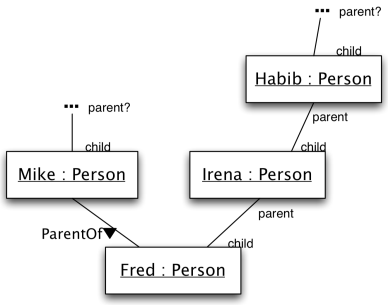
- A class can have an association with itself…



- Self associations have subtle semantics.
  - The example above requires that we always record two parents for every person in our system.
  - At some point we will not know someone's parent.

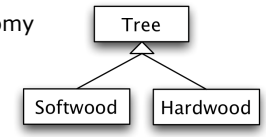
Class diagrams - Self association

- An object diagram shows the difficulty of trying to enforce mandatory self associations.
- The Parent cardinality should be 0..2 to reflect that fact that a person's parents may not be known.



Generalisation/Specialisation/Inheritance

- Represents a particular ordering/taxonomy
  - How do you classify trees?
    - Softwood/Hardwood?
    - Deciduous/Evergreen?

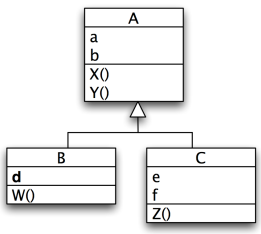


  - Neither classification is correct or incorrect. How you classify depends on the problem being solved.
  - We use generalisation to highlight common features in two or more classes.
  - Common features could be
    - Attributes
    - Behaviour (more common reason for generalisation)

Generalisation/Specialisation/Inheritance

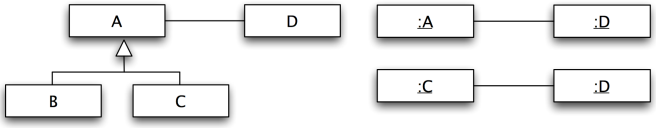
- The parent class contains features common to all subclasses
  - These include attributes, methods, associations

Class	Attributes	Methods
A	a, b	X, Y
B	a, b, d	X, Y, W
C	a, b, e, f	X, Y, Z



Generalisation

- Inheritance also extends to associations.
- If a parent class can take part in an association, then child classes can also take part in the same associations.
- The object diagrams (below right) are valid.



Draw class diagrams for...

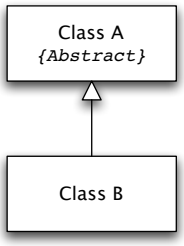
- A company or a person can own property.
- Both companies and people have a registered address.
- Companies have an ACN (Australian Company Number)
- Both have a TFN (Tax File Number)

Abstract Classes

- Generalisation (polymorphism and inheritance) is the key to reusability and extensibility, arguably the major benefits of OO.
- A goal of good OO design is to identify the general case (not easy!) and abstract it into a superclass.
- This leads to the idea of an abstract class
  - Instances of an abstract class are never created.
  - The class serves as a common "starting point" for other classes.
  - Only descendant non abstract classes are instantiated.
  - An abstract class can never appear in an object diagram.

Abstract Classes

- A class is specified as abstract by either:
  - writing the class name in italics
  - using the word {abstract} in the class definition.
- The diagram below uses both; normally only one notation would be used.

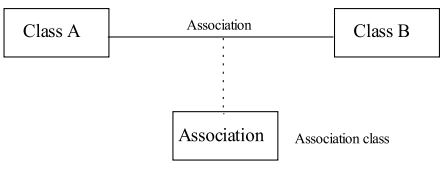


Attributed Associations

- Consider the following problem…
  - A contractor works for a several different companies.
  - A company employs many different contractors.
  - Where do you store the start/end dates of the contract?
  - They can't go in the contractor - there is no information about any specific companies.
  - Similarly for the company.
- The information is only needed when a relationship exists. These are details that belong to the relationship.

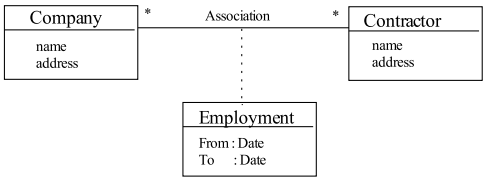
Attributed Associations

- Attributed associations are shown as…



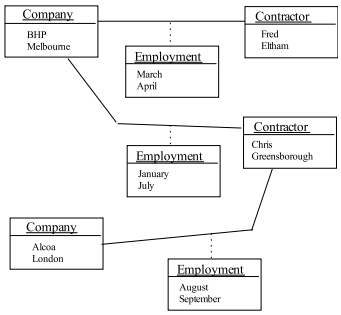
Attributed Associations

- An example class diagram…



Attributed Associations

- An object diagram:



Pop quiz questions

- An airline owns many planes which are flown by pilots

- A computer consists of a monitor, CPU, and a hard disk

Conventions for class diagrams

- Class-Name: Each class is unique within a system.
- Attributes: A data value held by objects in a class. Each attribute has a value for each object instance.
- Operations and Methods: An operation is a function or transformation that may be applied to or by objects in the class. All objects in the class share the same operations. A method is the implementation of an operation for the class.

Attributes and methods

- After you have decided what the objects are and how they relate to each other, think about what data needs to be stored about each object and what each object needs to be able to do.
- This will give you the attributes and methods for a class.

OOD example

- RMIT has decided they would like a new object oriented application which will keep track of students, staff, courses, and programs.

OOD example - 2

- RMIT employs staff who can be academic, administrative, or technical.
- Each staff member has a staff number and a name.
- Staff work for a department. Each department belongs to a faculty. RMIT has lots of each of these.
- Each department has a name and a department code.

OOD example - 3

- Departments run academic programs which consist of courses. Programs and courses each have a code and a name. Courses have a course guide.
- Students are enrolled in a program and take courses each semester. Students have a name, a student number, and a transcript which consists of all the marks they have received for different courses.

Questions:

- What objects do we need?
- What are the relationships between these objects?
- What data do we need to store about these objects?
- What functionality is required?

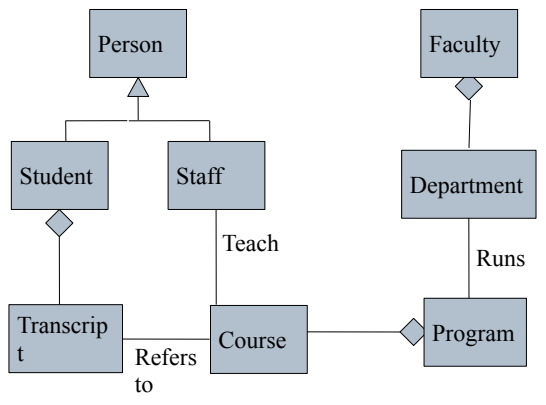
Possible Objects

- Staff
- Student
- Transcript
- Course
- Program
- Faculty
- Department

Relationships

- Staff and students are both types of people - they both have name, address, tax file number. Perhaps we can use inheritance?
- Students have transcripts (or perhaps a transcript is part of a student)
- Faculties consist of Departments
- Departments run Programs
- Programs consist of Subjects

One possible solution



Fleshing out the solution

- The next step would be to flesh this out with attributes (variables) and methods (functions).
- Each function then needs to be designed
- We should also document all the classes, methods, and attributes, in a data dictionary